CONTROL DATA CORPORATION

Development Division - Applications

DISK ROUTINES AND OVERLAYS

Chippewa Operating System

11/1/65

Disk Routines and Overlays

Contents

DISK ROUTINES AND OVERLAYS

Introduction

In the Chippewa Operating System, there is no single system element used to perform disk operations for all other elements of the system. Instead, each system element performs its own disk operations. This, while requiring additional coding for each of the system elements using the disk, eliminates the need for a request queueing and priority scheme required by the use of a single system element to process all disk operations. In addition, the housekeeping required by a disk subroutine in one system element can overlap, to some extent, a disk operation being performed by another system element. Among the system elements which perform disk operations are:

- peripheral processor resident (reads transient programs from the disk library)
- MTR (writes the contents of the dayfile buffer to the disk)
- some transient programs (read overlays from the disk)

Disk operations for external users are performed via the overlays 2WD (write disk), 2RD (read disk), and 2BD (backspace disk). These overlays are called by CIO when a disk operation is requested by a central processor program. In addition, these overlays are used by certain transient programs to perform disk operations. Thus, 1LJ and 1LT call 2WD when loading jobs from the card reader and a tape unit, respectively, while 1DJ and 1TD call 2RD when transferring job output to the printer or a tape unit.

Regardless of where in the system they are performed, disk operations are similar: this discussion will therefore be limited to the overlays 2WD, 2RD, and 2BD. Before discussing these routines a short review of the physical characteristics of the 6603 disk file is in order.

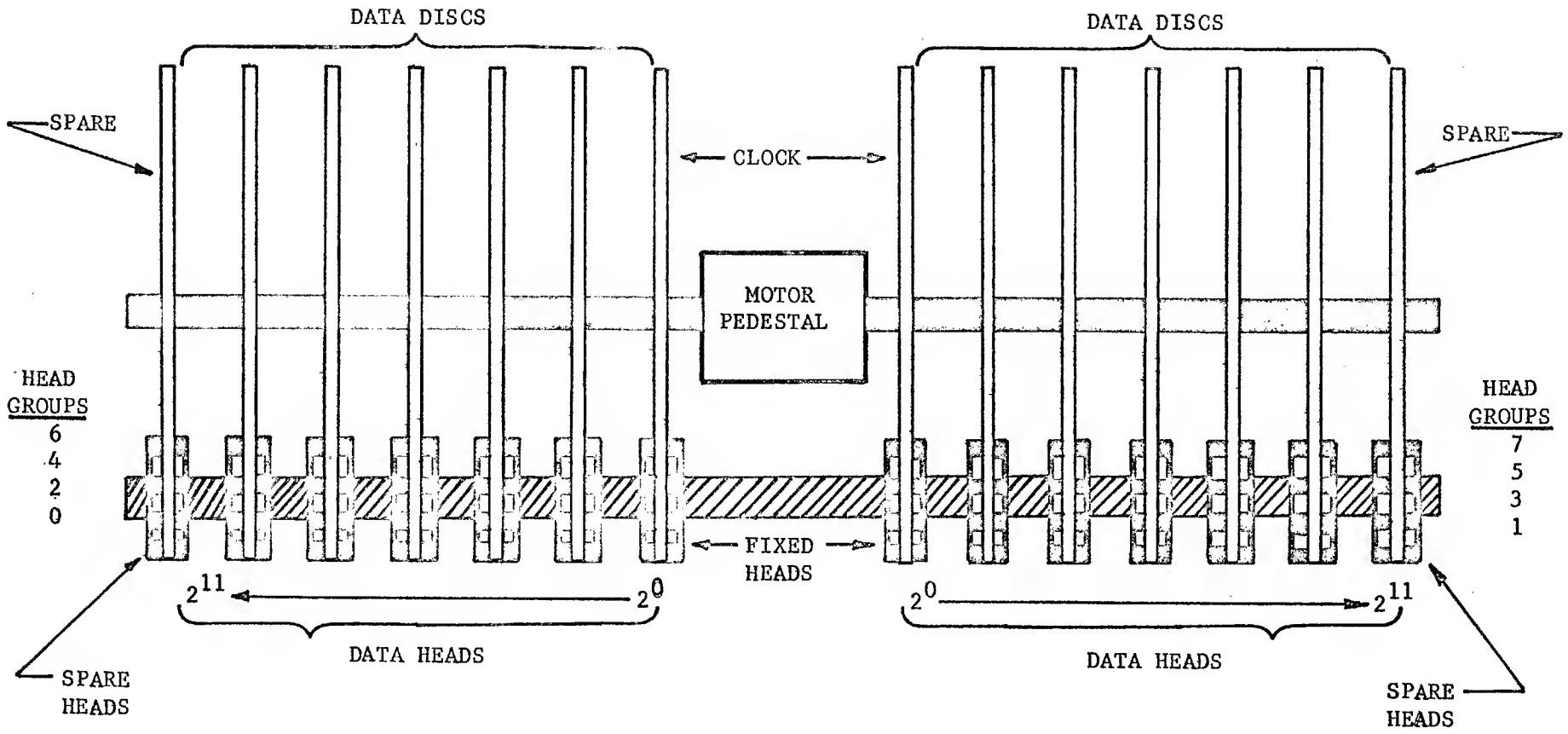
6603 Disk File: Description and Organization

The 6603 Disk File contains fourteen disks, each coated on both sides with magnetic oxide. Thus, there are a total of twenty-eight recording surfaces. On two of these surfaces timing tracks are recorded, two are used for spares, and twenty-four are used for recording data (see figure 1). All fourteen disks are mounted (in a vertical plane) on a common axis and rotate at a speed of approximately 900 revolutions per minute. Twelve of the data surfaces are on the right side of the unit, and twelve are on the left. Information is recorded on the disk in 12-bit bytes: each bit in a 12-bit byte is recorded on a separate disk surface.

Associated with each disk surface is a set of four read/write heads (see figure 2). An assembly consisting of a rocker arm and a head bar fits between each pair of facing disk surfaces. The head bar holds two sets of four heads, one set for each of the two facing surfaces. The read/write heads are mounted on this head bar in a fixed position relative to each other. The rocker arm-head bar assemblies for all disks mount on a common bracket which can be rotated. This rotation moves all the head bars simultaneously (with the exception of the heads accessing the timing track surfaces: these heads are fixed).

The disk surface is divided into four zones. A zone is that portion of the disk surface transversed by one of the four heads associated with that surface as the head (on its head bar-rocker arm assembly) moves through its maximum angular rotation. A byte may be written on the twelve data surfaces on the right side of the disk file or on the twelve data surfaces on the left side of the disk file: on either side, a byte may be written in any one of four zones. On each side of the disk file and for each zone on side, a single set of twelve read/write heads are used to record a byte (see figure 1). This set of twelve heads is called a head group. There are four head groups for each of the two sets of twelve disk surfaces: a total of eight head groups.

Each zone contains 128 tracks. A track is the recording path available to a given head group in a given position as the disk makes a complete revolution. To move from one track to another requires a physical



ALL HEADS (EXCEPT FIXED HEADS) MOVE TOGETHER

Figure 1

6603 DISC FILE

movement, or repositioning, of the head bar-rocker arm assemblies. At a given position, each head group accesses the same track in its zone. Thus, if head group 2 is positioned to track 125, the other 7 head groups are also positioned to track 125.

Tracks are divided into sectors: a sector is the smallest addressable segment of a track. There are 128 sectors in each of the tracks in the two outer zones. In the two innermost zones, there are only 100 sectors per track because of the reduced track length near the center of the disk compared to the track length available near the outside edge. A sector contains 351 bytes (each bit in a byte is recorded in one of 12 corresponding sectors across 12 disk surfaces). The first four bytes recorded are reserved for use by the controller: They provide a time lag between consecutive sectors and contain all zero bits. After the last data byte has been written, the controller writes a longitudinal parity byte. The sector format is illustrated in figure 3. Of the 351 bytes in a sector, then, five are used by the controller: The remaining 346 bytes may be used for data. Normally, 320 bytes (the equivalent of 64 central memory words) are used for data.

The number of words read from or written to the disk is solely a function of the word count specified in the IAM or OAM instruction. It is possible to read or write more than one sector at a time; it is possible to read or write in the group switch gap; it is possible for a read or write to wrap around on the same track. A read or write operation always begins at the beginning of a sector. When a write is initiated, the disk controller inserts four zero bytes before the data and inserts a parity byte after the last data byte. (The parity byte is not necessarily in the last byte position in a sector.) When a read is initiated, the controller assumes that the first four bytes are zero bytes, and does not pass these on to the data channel. When the word count in a read has been reduced to zero, the controller assumes that the next byte to be read is the parity byte. Thus, any attempt to read a number of bytes different than the number of bytes written will invariably create problems due to the interpretation of zero bytes and parity bytes as data and vice versa. For this reason, regardless of the amount of data to be recorded, a fixed number of bytes is written in each sector,

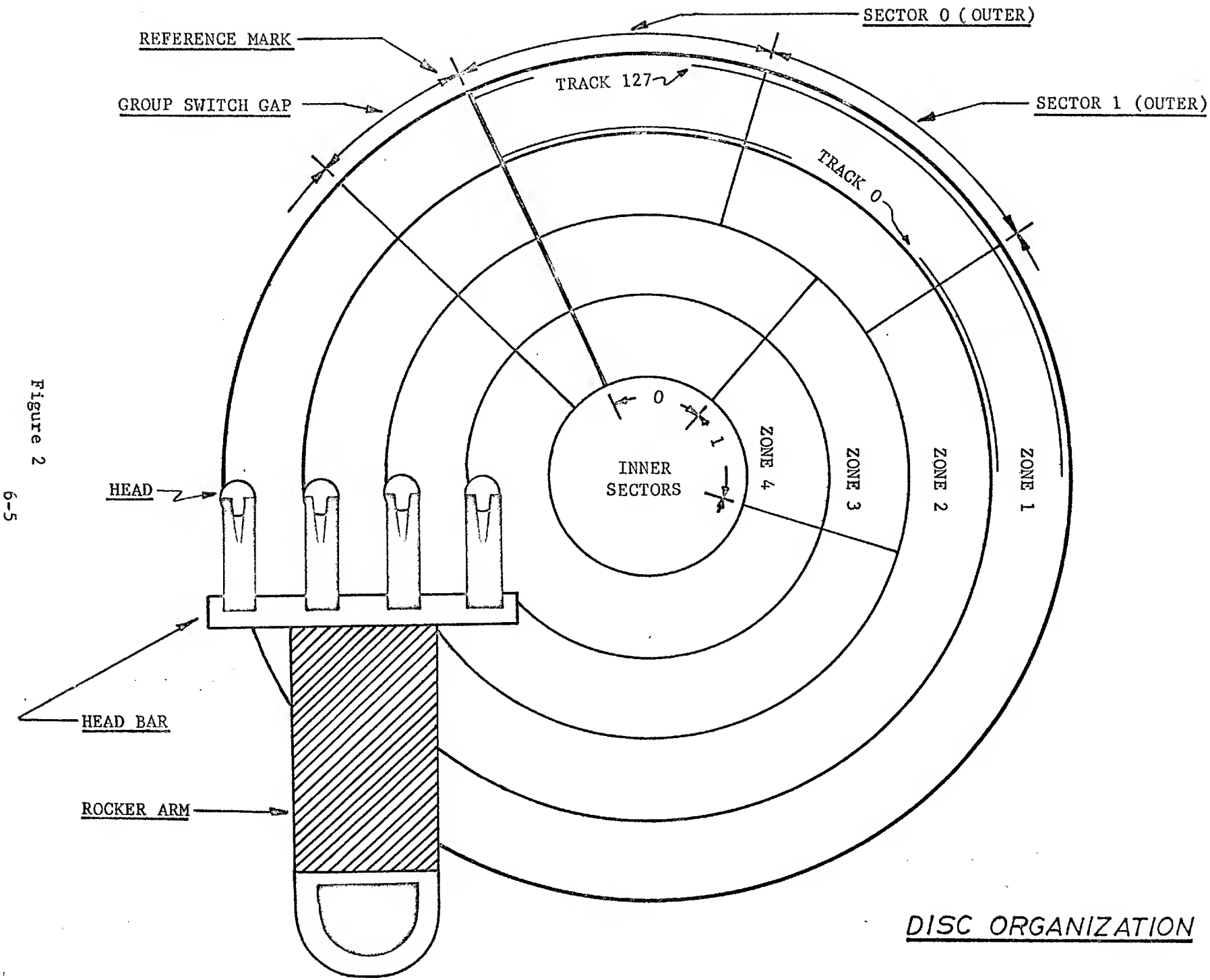


Figure 2

DISC ORGANIZATION

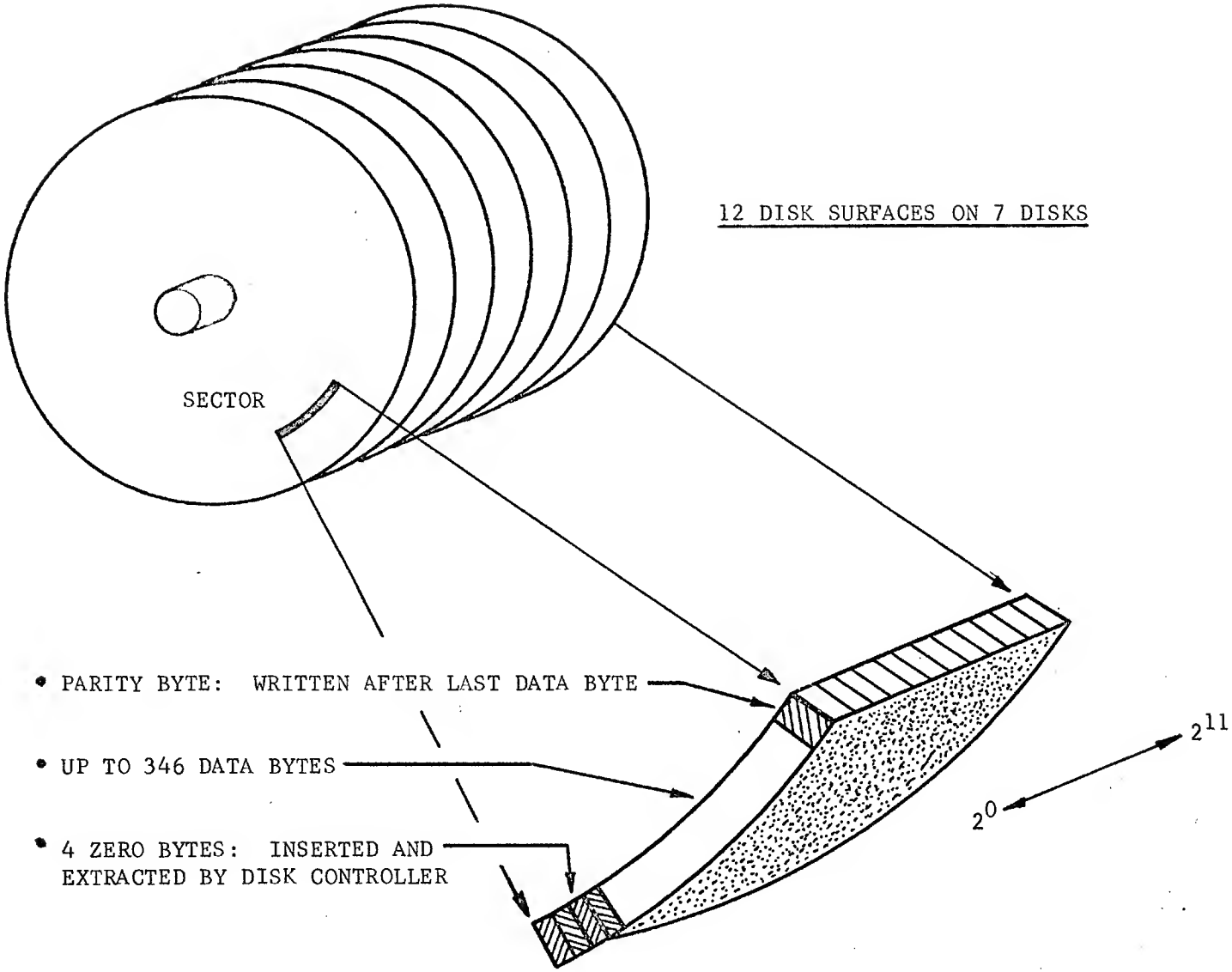


Figure 3

SECTOR FORMAT: 6603 DISK FILE

and only one sector is written at a time (i.e., data is recorded in physical records of one sector).

A reference mark on the disks containing the timing tracks defines the beginning of sector 0 in all four zones. Beyond this point, the starting point of sectors in the two inner zones does not coincide with the starting point of sectors in the two outer zones (see figure 2). The clock surfaces contain timing tracks for each zone. As the disk rotates, one of these timing tracks (depending on which head group is selected) drives a cell counter. This counter in turn triggers a sector counter. Both counters are initialized when the reference mark is detected. The cell counter is incremented as the timing track is read: When it reaches a count of 351, it is reset and the sector count advanced. The controller compares the sector number specified in a read or write function code: When equality is obtained, the read or write operation is initiated. The contents of the sector counter appear in the low-order 7 bits of the status response.

6603 Disk File: Timing Considerations

The rotational speed of the disk is approximately 900 revolutions per minute, corresponding to a revolution time of about 66 milliseconds. The time required to read or write a byte is approximately 1.4 microseconds on the two outer zones and 1.8 microseconds on the two inner zones. In the outer zones, then, a sector passes under the heads every 490 microseconds. It requires a minimum of 325 microseconds to transfer the 64 central memory words in a sector from peripheral processor memory to central memory, and, because of memory and pyramid conflicts, will probably require longer. A single peripheral processor cannot maintain a continuous data flow between consecutive sectors on the disk and central memory.

If the programmer wishes to read or write in a given sector, he simply issues the appropriate function code and, when the sector comes under the heads, the operation is initiated. The programmer may prefer to minimize the time spent waiting for this sector by sensing (via a status request) the position of the disk. Timing considerations make

it impossible to sense for a given sector and then initiate an operation in that sector: If one wishes to read or write sector N, then sector N-2 should be sensed in order to assure that a revolution will not be lost.

There are two types of delays which are of concern to the disk programmer. One of these is the positioning delay: The time required to move the heads to a new track. When a track select function has been received by the disk controller and positioning initiated, a delay determined by counting 4 reference marks is provided to permit the head assembly to stabilize. Thus, depending on when positioning is initiated, up to 264 milliseconds may be required. During positioning, a status request will receive a "NOT READY" reply.

The second type of delay is the switching delay encountered when a different head group is selected. When head group switching is initiated, the controller provides a one millisecond delay to allow the circuits to stabilize: Furthermore, reading or writing cannot be initiated until a reference mark is detected. Thus, depending on when the head group select function is issued, up to 66 milliseconds may be required for head group selection.

Between the last sector in a track (sector 127 in the outer zones, sector 99 in the inner zones) and the first sector (sector 0) on that track is an area called the group switch gap (see figure 2). This area is approximately equivalent to three sectors in size. It is provided to accommodate the minimum 1 millisecond switching delay. A programmer can thus read or write the last sector in a track, select a new head group, and read or write sector zero of the new track without incurring a delay.

The function code for head group selection is 160X, where X is the head group number (0-7). It is possible to vary the second octal digit in this function code (normally zero) from 1 to 7: In doing so, the manner in which the data signals from the disk are sampled is varied. Use of the feature is reserved for error routines.

6603 Disk File: Data Capacity

There are 128 physical positions of the heads: At any one position, a track may be accessed by selecting one of eight head groups. Thus, the disk has a total of 8 x 128 = 1024 tracks. Of the eight head groups, four cover inner zones and four cover outer zones. In the inner zones, there are 100 sectors per track: In the outer zones, there are 128 sectors per track. Therefore, 512 tracks each contain 100 sectors while the other 512 tracks each contain 128 sectors. The disk file thus contains 116, 736 sectors. In normal use, up to 64 central memory words are recorded in a sector. The capacity of the 6603 disk file is thus approximately 7.5 million central memory words.

Chippewa Operating System Disk Usage

As we have seen, a single peripheral processor cannot maintain a continuous data flow from consecutive disk sectors to central memory. Therefore, the Chippewa Operating System uses a half track scheme in its disk operations. A half track is composed of either the odd-numbered or the even-numbered sectors in a track. In a disk operation, the system reads or writes alternate sectors, transferring data to or from central memory while passing over the intervening sector. Since the disk contains 1024 physical tracks, the equivalent half track capacity is 2048. The allocation of half tracks is controlled by MTR: disk write routines obtain half track addresses from MTR via the Request Track function. MTR maintains a table called the Track Reservation Table (TRT) which contains an entry for each half track on a disk. On receipt of the Request Track function, MTR searches the table for an unassigned half track, and returns the half track address to the requestor in the upper byte of the Message Buffer. If no half track is available, a zero address is returned to the requestor. A half track is never split between files: thus, the half track is the smallest unit of storage allocated on the disk.

The format of the half track address, and its relationship to physical disk addresses, is illustrated below.

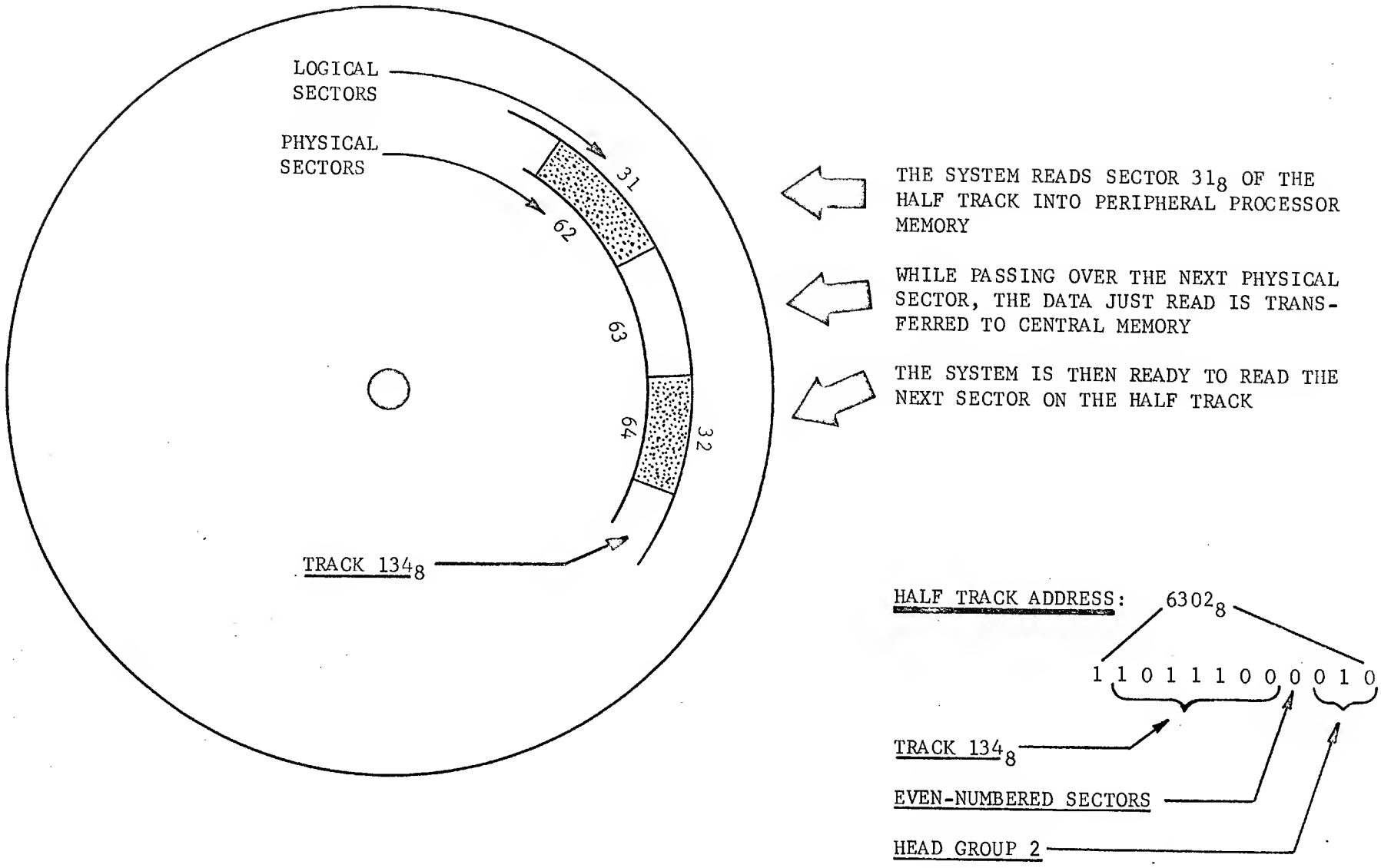
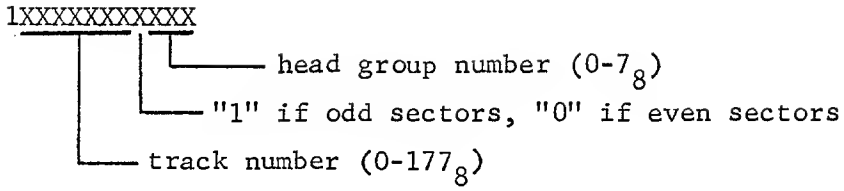


Figure 4

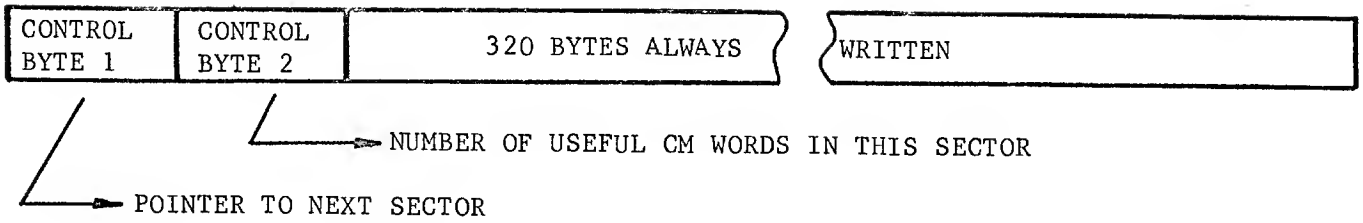
HALF TRACK USE: AN EXAMPLE



Sector numbers maintained by the system (such as the Current Sector in an FST entry) are logical sector numbers, and refer to a sector within a half track. In the outer zones, sectors within a half track are numbered $0\text{-}77_8$: In the inner zones, sectors within a half track are numbered $0\text{-}61_8$. To convert a logical sector number to a physical sector number, the system shifts the logical sector number left one place and inserts the 2^4 bit from the half track address into the low-order bit position. For example, consider logical sector 77_8 (63_{10}) in a half track composed of the odd-numbered sectors in a physical track. In this case, the 2^4 bit of the half track address will be a "1". By shifting the logical sector left one place and inserting the "1" bit from the 2^4 bit position of the half track address, we obtain 177_8 (127_{10}) for the physical sector number. For the remainder of our discussion, a reference to "sector number" will refer to the logical sector number unless otherwise described.

For files recorded on the disk, the physical record is, of course, the sector. A logical record may be composed of several sectors. The format of the physical record is shown in figure 5. 502_8 bytes are always written in each sector. The first two bytes written are control bytes: the remaining 500_8 bytes are data bytes. Control byte 2 contains the number of useful central memory words in this sector: If control byte 2 contains 100_8, all 500_8 bytes in this sector contain useful information. A sector in which control byte 2 contains less than 100_8 is called a short sector, and is interpreted as a record mark. A logical record may comprise several full sectors, but is always terminated by a short sector. If the data to be recorded as a logical record is a multiple of 100_8 CM words, the system will write, as the record mark, a sector in which control byte 2 contains zero.

Control byte one points to the next physical record in this file. If the next sector is on the same half track, then this byte contains the



CONTROL BYTE 1	CONTROL BYTE 2	RECORD
NON-ZERO	100_8	"FULL" SECTOR: PART OF A LOGICAL RECORD
NON-ZERO	NON-ZERO, $<100_8$	"SHORT" SECTOR: PART OF A LOGICAL RECORD; RECORD MARK
NON-ZERO	ZERO	"SHORT" SECTOR: RECORD MARK
ZERO	ZERO	FILE MARK

Figure 5

DISK FILE PHYSICAL RECORD FORMAT

number of that sector. If the next sector is on another half track, then this byte contains the half track address for that half track. (The file would be continued beginning with sector zero of the new half track.)

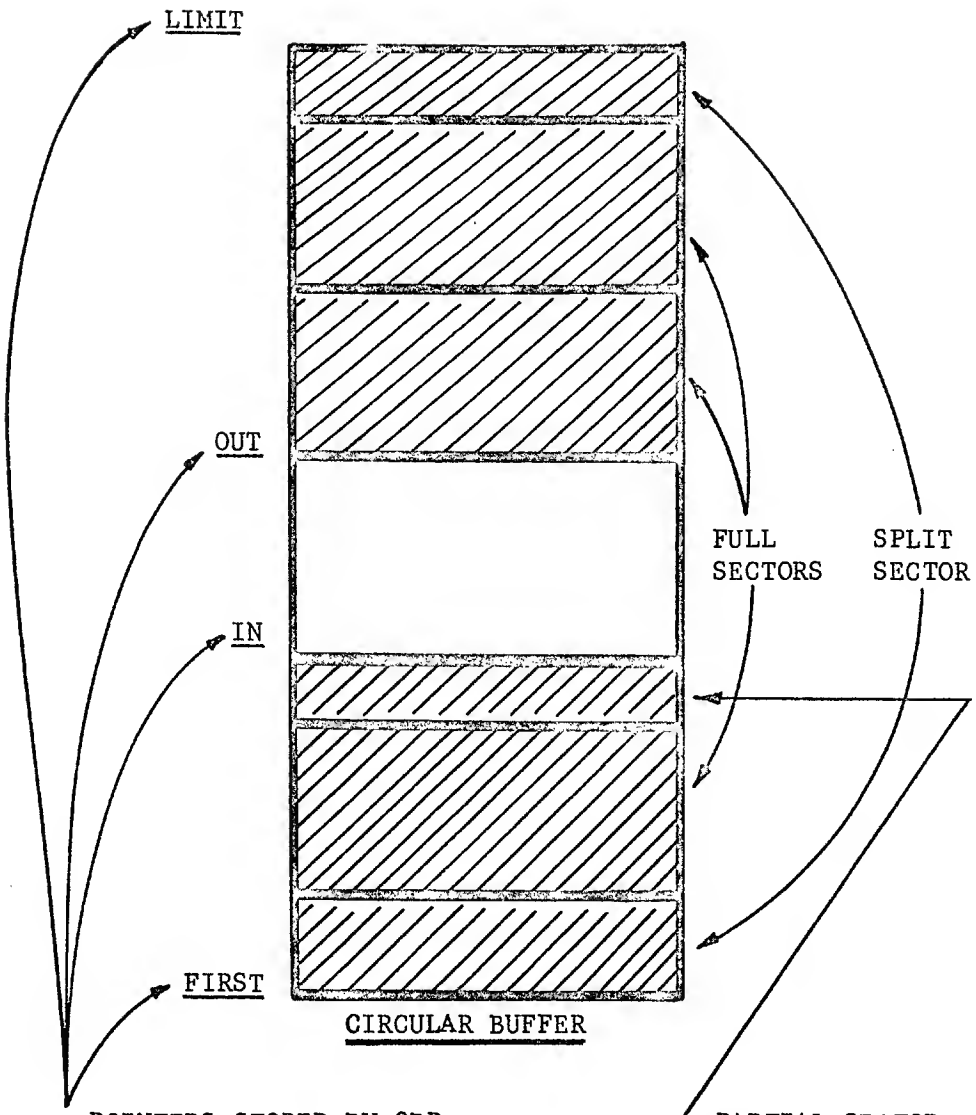
At the end of each write operation, the system writes a file mark. The Current Sector byte of the FST entry is not incremented to reflect this file mark sector, so the effect is equivalent to writing a file mark and backspacing over it. On the disk, a file mark is a sector in which both control bytes contain zero.

The Disk Write Overlay, 2WD

Disk write requests by users are executed by CIO's overlay 2WD. This overlay is also used by 1LJ and 1LT in loading jobs on the disk. Before calling 2WD, CIO calls the 2BP overlay to check the legality of the buffer parameters FIRST, IN, OUT, and LIMIT. After checking these parameters, 2BP searches the File Name Table for the file name specified in the CIO call (i.e., in the first word of the argument list). When found, 2BP stores the address of the corresponding FST entry. Should the file name not be found in the FNT, 2BP constructs an FNT entry for this file. Finally, 2BP clears the 2^0 bit in the buffer status byte of the FST entry to reserve the file.

CIO then calls 2WD. (Refer to the flow chart on page A-1.) 2WD reads the FST entry for the file and extracts the equipment number from byte one. The equipment number is added to the EST base address, and the EST entry read. The channel number from byte 2 of the EST entry is then inserted in the appropriate I/O instructions.

The output data in the circular buffer may appear as a contiguous block, or may wrap around the buffer, as illustrated in figure 6. In computing the total number of sectors in the circular buffer, then, the 2WD routine first subtracts OUT from IN. If the difference is positive, then this difference is the total number of words to be written, and 2WD shifts off the lower six bits of this word count in order to obtain the equivalent number of sectors. If OUT-IN is negative, the value of LIMIT is added to the difference and FIRST subtracted to obtain the



BUFFER PARAMETER PROCESSING

1. COMPUTE TOTAL NUMBER OF WORDS IN OUTPUT AREA
2. COMPUTE TOTAL NUMBER OF SECTORS IN OUTPUT AREA BY SHIFTING TOTAL WORD COUNT RIGHT 6 PLACES
3. COMPUTE NUMBER OF WORDS BETWEEN OUT AND LIMIT
4. COMPUTE NUMBER OF SECTORS BETWEEN OUT AND LIMIT BY SHIFTING OUT-LIMIT WORD COUNT RIGHT 6 PLACES
5. EXTRACT LOW-ORDER 6 BITS OF OUT-LIMIT WORD COUNT: THIS GIVES THE NUMBER OF WORDS IN THAT PART OF THE SPLIT SECTOR BETWEEN OUT AND LIMIT
6. SUBTRACT NUMBER OF WORDS COMPUTED IN (5) FROM 100_8 TO GET NUMBER OF WORDS TO BE READ FROM THE BUFFER BEGINNING AT FIRST IN ORDER TO COMPLETE THE SPLIT SECTOR
7. SET UP INSTRUCTIONS FOR PROCESSING THE SPLIT SECTOR

Figure 6

CIRCULAR BUFFER PARAMETER PROCESSING - 2WD OVERLAY

total word count and, from that, the equivalent number of sectors.

Regardless of whether the data is contiguous or wraps around the buffer, 2WD proceeds on the assumption that the data does wrap around, and proceeds to compute the values needed to process the wraparound case. The steps involved are listed in figure 6. These values, although always computed, are not required in the contiguous case: in either case, the terminal path is entered when the total sector count is reduced to zero. By computing these values regardless of whether the data is contiguous in the buffer or wraps around the buffer, computations during the period when the disk is actively in use are reduced.

Next, 2WD picks up the channel number from the EST entry and requests reservation of that channel from MTR. The Current Track byte of the FST entry for this file is then examined. If this byte is zero, then this file has not previously been used. A half track assignment is requested from MTR: MTR returns a half track address to the requestor in byte one of the first word in the message buffer. If no half track is available, MTR will return a zero byte to the requestor: 2WD then inserts an error message in the dayfile and aborts the control point after dropping the channel reservation. 2WD now has the address of the half track where the next operation is to be performed, and proceeds to position the disk to this half track. This half track address is compared with byte 2 of the TRT pointer word for this disk, and repositioning or head group selection performed only if required. Byte 2 of the TRT pointer is then updated.

2WD next requests another half track assignment from MTR. This half track is a spare: by keeping it available, it is possible for 2WD to switch head groups within the group switch gap if this action should be required when the end of the current half track is reached.

The transfer of data from the buffer to the disk then begins. 2WD reads 100_8 words from central memory into peripheral processor memory, sets control bytes one and two, and then writes the completed sector to the disk. As each sector is written, the number of the sector is examined to determine if the end of the half track is reached. To do this, 2WD compares the sector number with byte 4 of the TRT pointer word (if head

group number = 0-3) or byte 5 of the TRT pointer word (if head group number = 4-7). These bytes contain the values 100_8 and 62_8, respectively.

If the end of the half track has been reached, 2WD positions the disk to the spare half track: again, the half track address is compared with byte 2 of the TRT pointer word and positioning or head group selection performed only if required. After initiating any repositioning which might be required, 2WD requests a spare half track from MTR.

2WD continues reading 100_8-word blocks from central memory and writing them to the disk until it recognizes that there is not enough data in the circular buffer for a complete sector. (Some part of a sector may still, however, remain.) 2WD then examines the buffer status contained in byte 5 of the FST entry to see if an end record was requested (2^4 bit = 1). If an end record was requested, 2WD writes a short sector to the disk. If any data remained in the circular buffer, it will be written in this short sector: otherwise, control byte 2 will simply be set to zero.

After the last data sector has been written to the disk, 2WD writes a file mark - a sector with both control bytes equal to zero. The Current Sector byte of the FST entry is not, however, incremented to reflect the writing of this file mark: the next write to this file will write over the file mark sector. After the file mark has been written, 2WD requests MTR to drop the spare half track assignment and to release the channel reservation.

If no end record function was requested, 2WD simply updates the OUT pointer before returning control to CIO: There may still be some data in the circular buffer. If an end record function was requested, no data remains in the buffer: 2WD therefore sets IN = OUT = FIRST to indicate that the buffer is empty.

When control is returned to CIO, CIO sets the 2^0 bit of the buffer status in the FST entry to 1 to indicate that the file is no longer in use, and sets the 2^0 bit of the buffer status in the calling program's argument list to 1 to indicate to the calling program that the operation has been completed.

The Disk Read Overlay, 2RD

Disk read requests by users are executed by CIO's overlay 2RD. This overlay is also used by 1DJ and 1TD. The processing performed by 2BP in this case is identical to that performed in the case of 2WD. On entry, 2RD reads the FST entry for the file, picks up the equipment number from byte one, and uses this number to obtain the EST entry. The channel number from the EST entry is then set in the I/O instructions.

2RD then proceeds to compute the number of sectors which can be loaded into the circular buffer. If there is not room for a full sector, control is returned to CIO. The data to be read may fit in the buffer in a contiguous block, or may wrap around the buffer. The computation of the values (total word count, total sector count, etc.) used in controlling the transfer of data to the buffer is performed in a manner similar to 2WD. Again, the wraparound case is assumed.

The Current Track byte of the FST entry is examined. If this byte is zero, the file has not been used before and so contains no data. 2RD sets the buffer status to indicate a file mark and returns control to CIO.

2RD requests a channel reservation from MTR and positions the disk to the half track address contained in the FST entry's Current Track byte. As in all disk routines, the half track address is compared with the disk position specified in the TRT pointer, and repositioning or head group switching performed only if necessary.

2RD then uses the Current Sector byte of the FST entry to construct the read function code, and reads the specified sector into peripheral processor memory. A status request is then issued, and the response is examined to determine if a parity error occurred. In the event of a parity error, the system rereads the sector three times; once using the normal sampling method and twice at
varied sampling margins. If the parity error re-occurs in each of the rereads, 2RD inserts an error message in the dayfile and stops (via a UJN 0 instruction). Since the halt occurs without the disk channel being released, all system activity will shortly cease (if this disk is the

system disk, disk 0). A dead start load will be necessary to reinitiate processing.

If the read was successful, 2RD examines the high-order six bits of control byte one: if these bits are zero, then this control byte contains a sector number, while if these bits are non-zero, this control byte contains a half track number. In the latter case, 2RD positions the disk to the new half track address. While any repositioning or head group switching which might be required is in process, 2RD transfers the number of words specified in control byte 2 from peripheral processor memory to the circular buffer, and updates the values used in controlling the transfer. If the sector just read was a full sector (100_8 CM words of data), and if there is enough room in the circular buffer for another full sector, 2RD loops to read the next sector from the disk.

If the last sector read was a short sector, then the end of a logical record has been reached, and the buffer status is set to reflect a record mark. If the end of logical record has been reached, or if there is not enough room in the circular buffer for a full sector, 2RD requests MTR to release the channel reservation, updates the IN pointer in the calling program's argument list, and returns control to CIO. CIO updates the buffer status in the FST entry to release the file reservation, and updates the buffer status in the calling program's argument list to indicate that the operation has been completed.

If, after reading the last logical record in a file, the calling program issues another read to the file, the file mark will be read. The processing proceeds as described above: 2RD reads a sector whose address is specified in the Current Track and Current Sector bytes of the FST entry. Since control byte 2 is zero, 2RD recognizes this as a short sector, sets the buffer status to reflect a record mark, and releases the channel. 2RD then examines control byte one; since this contains zero, the file mark is recognized and the buffer status set accordingly before returning control to CIO.

The Backspace Disk Overlay, 2BD

Disk backspacing may take the form of a BCD backspace or, more commonly, a binary backspace. In either case, it is desired to backspace over a logical record, and it is assumed that any backspacing over logical records in the buffer has been done by the calling program. Backspacing over the physical records which may constitute a logical record is essentially a matter of backspacing over two sectors and then reading a sector.

2BD uses a subroutine to backspace over a sector. (See flow chart on page A-5.) This subroutine examines the Current Sector byte of the FST entry, and, if non-zero, subtracts one from this number and exits. This is equivalent to backspacing over one physical record (i.e., one sector). If the Current Sector number is zero, then the preceding physical record is on another half track. In this case, the subroutine stores the Current Track byte from the EST entry for this file, since it will have to search the file for a sector which has this half track address contained in control byte one.

The subroutine rewinds the file by picking up the Beginning Track byte from the FST entry. (Should the Beginning Track byte be equal to the Current Track byte, the subroutine exits, since this indicates that the system has backspaced over all physical records in this file.) After rewinding the file, the subroutine reads each sector in the file until it finds a sector with the desired half track address in control byte one. The number of this sector is then stored, and control returned to the calling routine. A backspace operation on a file of any size may take considerable time if it should become necessary to rewind the file and search forward.

A binary backspace on the disk consists of backspacing over two sectors (using the subroutine described above) and reading a sector until a short record is found, indicating the end of a logical record. 2BD sets the circular buffer pointers IN and OUT equal to FIRST, and returns control to CIO. CIO updates the buffer status in the FST entry and in the calling program's argument list before exiting.

It is also possible to issue a BCD backspace to the disk. For the disk, as for 1" tape (but not for 1" tape), a logical BCD record consists of a series of central memory words presumably containing display code data, terminated by a central memory whose low-order byte (byte 5) is zero.

The BCD backspace begins with the computation of the amount of data left in the buffer as a result of the last read. This quantity, referred to as D, is equal in IN-OUT if the data in the buffer is contiguous, or IN-OUT + LIMIT-FIRST if the data wraps around the buffer. This data was left in the buffer as a result of the last read, and may have been stored on the disk in several sectors. The system assumes that the calling program will backspace within the buffer, and so, before beginning a logical BCD record backspace on the disk, 2BD will backspace the disk a number of sectors equivalent to the amount of data contained in the buffer. This quantity is represented by D.

2BD therefore backspaces over a sector (by the same subroutine used in binary backspacing and described earlier) and reads that sector into peripheral processor memory. The sector length in control byte 2 is then compared with D: if less than D, then this sector is assumed to contain data which has already been read into the buffer. 2BD then decreases D by this amount, backspaces over this sector and the sector preceding it, and then reads a sector. The process of backspacing, reading, and reducing D is repeated until a sector is read whose length is greater than the present value of D: this sector could not entirely be part of the read data in the buffer, and so must be searched for a logical record. 2BD transfers this sector from peripheral processor memory to the circular buffer beginning at FIRST. If D is still non-zero, then part of this sector contains data residing in the buffer at the time the backspace was requested, and presumably has been searched by the calling program: 2BD therefore sets the OUT pointer to FIRST + sector length - D. At the same time, the IN pointer is set to reflect the transfer of the sector to the buffer.

2BD then searches each word in the buffer from OUT - 1 down to FIRST until a word with a zero low-order byte is found, indicating the end of a logical BCD record. When the end of the record is found, 2BD updates the IN and OUT pointers in the calling program's argument list, and

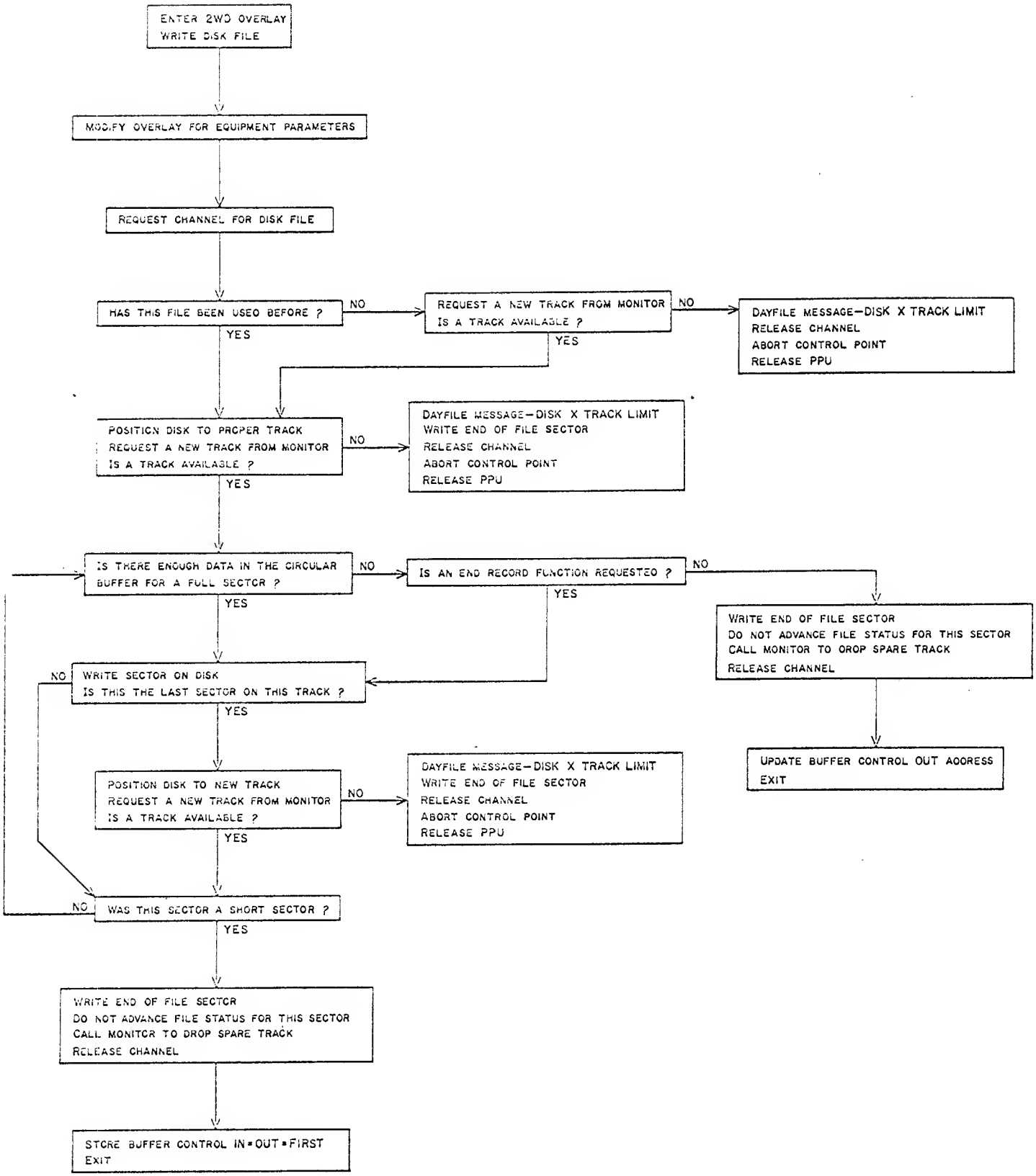
returns control to CIO. OUT now points to the first word following the end of the logical record. If no zero low-order byte was found, then 2BD backspaces two sectors and reads one, and then repeats the buffer search.

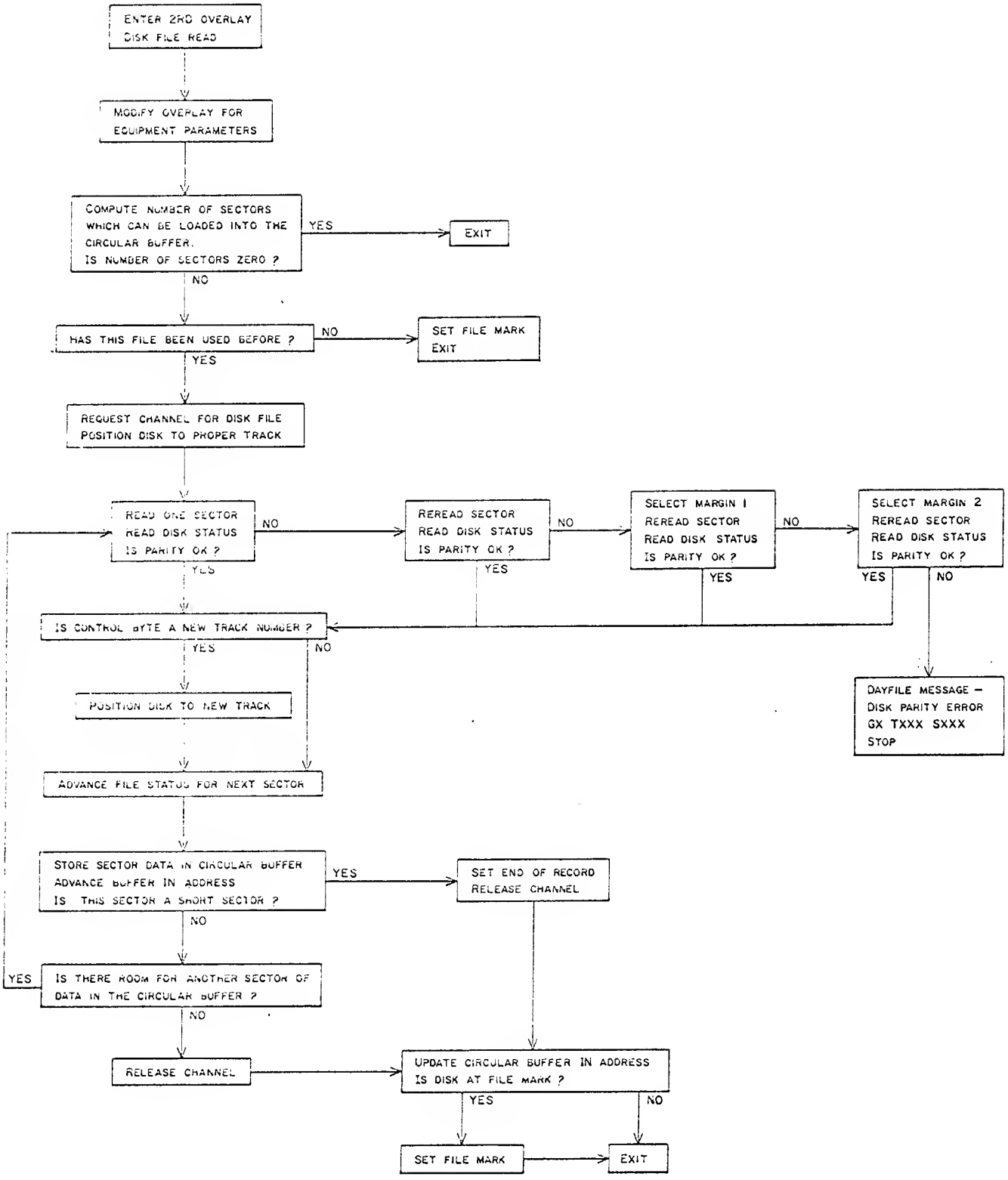
The Drop Track Overlay, 2DT

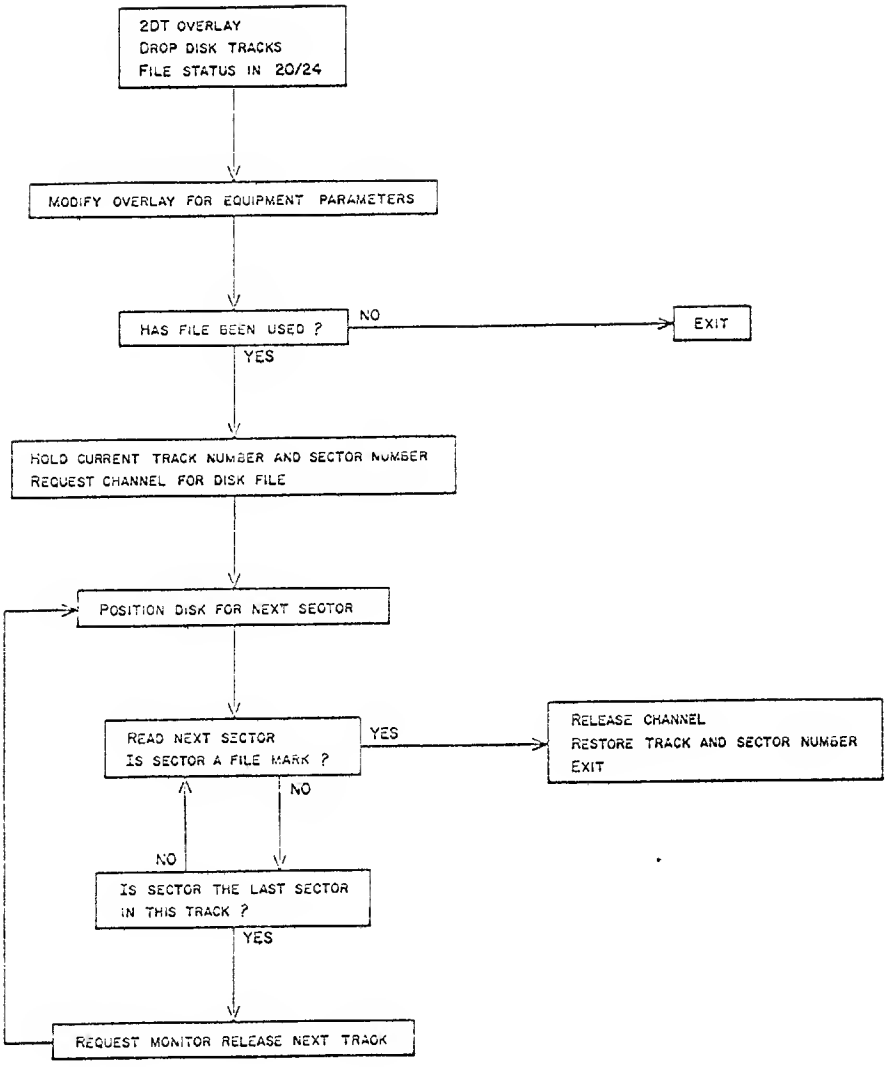
When CIO receives a disk write request, it first calls the 2BP overlay to check the legality of the buffer parameters and to search the FNT for the file name. CIO then reads the EST entry for this file, and examines the buffer status in byte 5. If the buffer status indicates that the last operation performed on this file was a read operation, then an overlay, 2DT, is called to drop the subsequent portion of the file. In effect, then, if some part of a file is read and it is then decided to write to that file, the remainder of the file is erased.

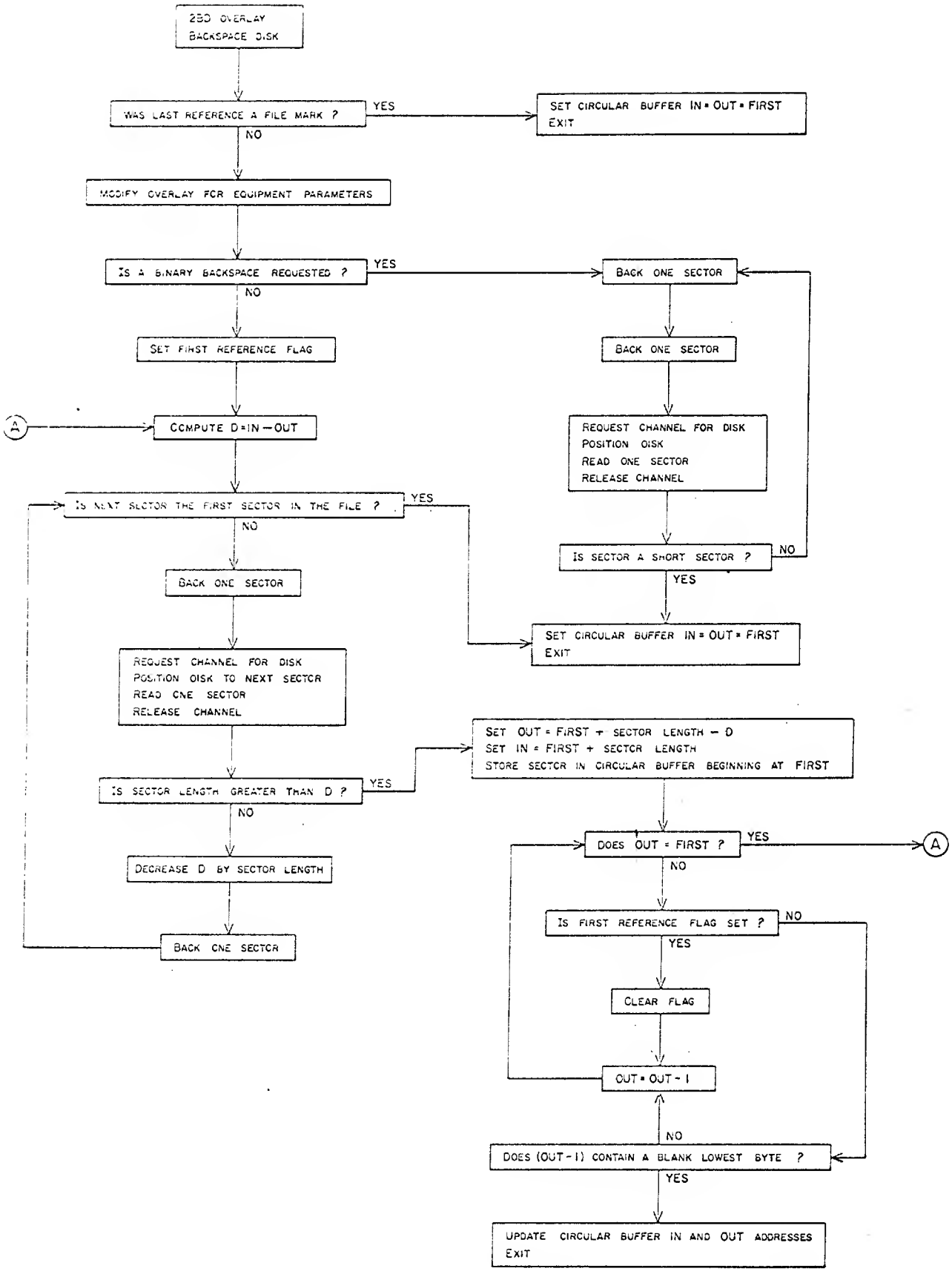
The flow chart for the 2DT overlay is shown on page A-3 of the attached flow charts. The routine picks up the Current Track byte and Current Sector byte from the FST entry for the file, and reads the sector at this address. If this sector is a file mark, 2DT returns control to CIO. If control byte one of this sector contains a half track address, 2DT requests MTR to drop this half track reservation. MTR then clears the bit in the Track Reservation Table corresponding to this half track address. 2DT positions the disk to this half track address and begins reading sectors until a file mark is found or the end of the half track is reached. The process of reading and dropping half tracks continues until the end of the file is reached.

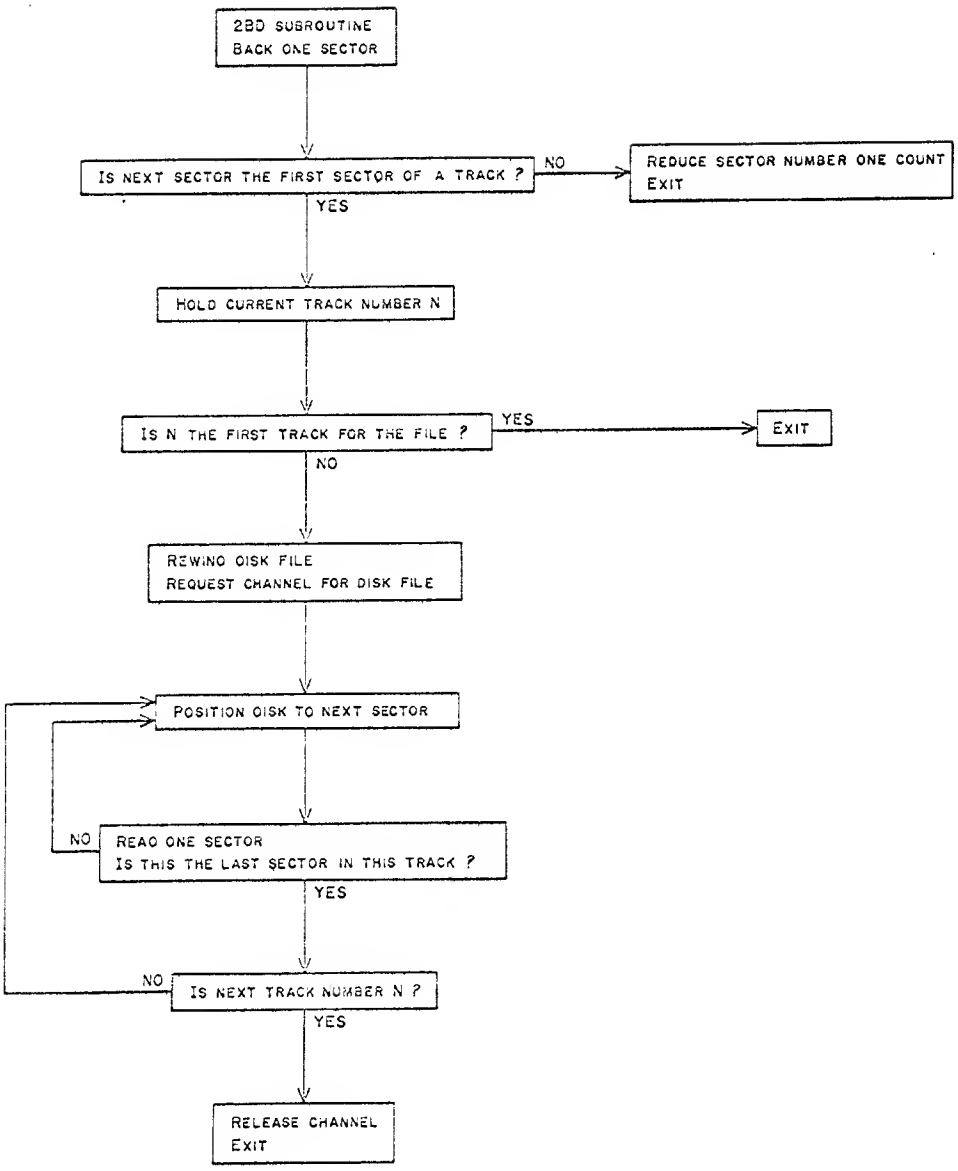
At the end of a job, all local files associated with the job are dropped. For disk files, a process similar to that described above is required to release half track reservations. This is performed for 1AJ by the 2DF overlay. 2DF differs from 2DT in that 2DF drops files assigned to other equipment as well as those assigned to the disk, and 2DF drops all the half tracks reserved by a file, not just those following the half track specified in the Current Track byte of the FST entry. 2DF is also called by 1DJ and 1TD when printing files or writing files on tape.


ENTER 2WD OVERLAY
WRITE DISK FILE
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
REQUEST CHANNEL FOR DISK FILE
HAS THIS FILE BEEN USED BEFORE ?
NO
YES
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
YES
DAYFILE MESSAGE—DISK X TRACK LIMIT
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
POSITION DISK TO PROPER TRACK
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
YES
DAYFILE MESSAGE—DISK X TRACK LIMIT
WRITE END OF FILE SECTOR
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
IS THERE ENOUGH DATA IN THE CIRCULAR BUFFER FOR A FULL SECTOR ?
NO
YES
IS AN END RECORD FUNCTION REQUESTED ?
NO
YES
WRITE END OF FILE SECTOR
DO NOT ADVANCE FILE STATUS FOR THIS SECTOR
CALL MONITOR TO DROP SPARE TRACK
RELEASE CHANNEL
UPDATE BUFFER CONTROL OUT ADDRESS
EXIT
NO
WRITE SECTOR ON DISK
IS THIS THE LAST SECTOR ON THIS TRACK ?
YES
POSITION DISK TO NEW TRACK
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
YES
DAYFILE MESSAGE—DISK X TRACK LIMIT
WRITE END OF FILE SECTOR
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
NO
WAS THIS SECTOR A SHORT SECTOR ?
YES
WRITE END OF FILE SECTOR
DO NOT ADVANCE FILE STATUS FOR THIS SECTOR
CALL MONITOR TO DROP SPARE TRACK
RELEASE CHANNEL
STORE BUFFER CONTROL IN=OUT=FIRST
EXIT


ENTER 2RD OVERLAY
DISK FILE READ
MODIFY OVERLAY FOR
EQUIPMENT PARAMETERS
COMPUTE NUMBER OF SECTORS
WHICH CAN BE LOADED INTO THE
CIRCULAR BUFFER.
IS NUMBER OF SECTORS ZERO ?
YES
EXIT
NO
HAS THIS FILE BEEN USED BEFORE ?
NO
SET FILE MARK
EXIT
YES
REQUEST CHANNEL FOR DISK FILE
POSITION DISK TO PROPER TRACK
READ ONE SECTOR
READ DISK STATUS
IS PARITY OK ?
NO
REREAD SECTOR
READ DISK STATUS
IS PARITY OK ?
NO
SELECT MARGIN 1
REREAD SECTOR
READ DISK STATUS
IS PARITY OK ?
NO
SELECT MARGIN 2
REREAD SECTOR
READ DISK STATUS
IS PARITY OK ?
YES
YES
YES
YES
NO
IS CONTROL BYTE A NEW TRACK NUMBER ?
YES
NO
DAYFILE MESSAGE –
DISK PARITY ERROR
GX TXXX SXXX
STOP
POSITION DISK TO NEW TRACK
ADVANCE FILE STATUS FOR NEXT SECTOR
STORE SECTOR DATA IN CIRCULAR BUFFER
ADVANCE BUFFER IN ADDRESS
IS THIS SECTOR A SHORT SECTOR ?
YES
SET END OF RECORD
RELEASE CHANNEL
NO
YES
IS THERE ROOM FOR ANOTHER SECTOR OF
DATA IN THE CIRCULAR BUFFER ?
NO
RELEASE CHANNEL
UPDATE CIRCULAR BUFFER IN ADDRESS
IS DISK AT FILE MARK ?
YES
NO
SET FILE MARK
EXIT


2DT OVERLAY
DROP DISK TRACKS
FILE STATUS IN 20/24
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
HAS FILE BEEN USED ?
NO
EXIT
YES
HOLD CURRENT TRACK NUMBER AND SECTOR NUMBER
REQUEST CHANNEL FOR DISK FILE
POSITION DISK FOR NEXT SECTOR
READ NEXT SECTOR
IS SECTOR A FILE MARK ?
YES
RELEASE CHANNEL
RESTORE TRACK AND SECTOR NUMBER
EXIT
NO
NO
IS SECTOR THE LAST SECTOR
IN THIS TRACK ?
YES
REQUEST MONITOR RELEASE NEXT TRACK


2B0 OVERLAY
BACKSPACE DISK
WAS LAST REFERENCE A FILE MARK ?
YES
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
NO
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
IS A BINARY BACKSPACE REQUESTED ?
YES
BACK ONE SECTOR
NO
SET FIRST REFERENCE FLAG
BACK ONE SECTOR
A
COMPUTE D = IN − OUT
REQUEST CHANNEL FOR DISK
POSITION DISK
READ ONE SECTOR
RELEASE CHANNEL
IS NEXT SECTOR THE FIRST SECTOR IN THE FILE ?
YES
NO
IS SECTOR A SHORT SECTOR ?
NO
YES
BACK ONE SECTOR
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
REQUEST CHANNEL FOR DISK
POSITION DISK TO NEXT SECTOR
READ ONE SECTOR
RELEASE CHANNEL
SET OUT = FIRST + SECTOR LENGTH − D
SET IN = FIRST + SECTOR LENGTH
STORE SECTOR IN CIRCULAR BUFFER BEGINNING AT FIRST
IS SECTOR LENGTH GREATER THAN D ?
YES
NO
DOES OUT = FIRST ?
YES
A
NO
DECREASE D BY SECTOR LENGTH
IS FIRST REFERENCE FLAG SET ?
NO
YES
BACK ONE SECTOR
CLEAR FLAG
OUT = OUT − 1
NO
DOES (OUT − 1) CONTAIN A BLANK LOWEST BYTE ?
YES
UPDATE CIRCULAR BUFFER IN AND OUT ADDRESSES
EXIT


2BD SUBROUTINE
BACK ONE SECTOR
IS NEXT SECTOR THE FIRST SECTOR OF A TRACK ?
NO
REDUCE SECTOR NUMBER ONE COUNT
EXIT
YES
HOLD CURRENT TRACK NUMBER N
IS N THE FIRST TRACK FOR THE FILE ?
YES
EXIT
NO
REWIND DISK FILE
REQUEST CHANNEL FOR DISK FILE
POSITION DISK TO NEXT SECTOR
NO
READ ONE SECTOR
IS THIS THE LAST SECTOR IN THIS TRACK ?
YES
NO
IS NEXT TRACK NUMBER N ?
YES
RELEASE CHANNEL
EXIT